Recommending software features to designers: From the perspective of users

Recommending Software Features to Designers: From the Perspective of Users

Journal:	*Software: Practice and Experience*
Manuscript ID	SPE-19-0183.R1
Wiley - Manuscript type:	Research Article
Date Submitted by the Author:	21-Jan-2020
Complete List of Authors:	liu, chun; Henan University Yang, Wei; Henan University, School of Software and Information Engineering Li, Zheng; Henan University, School of Software and Information Engineering Yu, Yijun; The Open University, Computing and Communications
Keywords:	feature extraction, feature recommendation, association rules, lasso regression

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Recommending Software Features to Designers: From the Perspective of Users

Chun Liu[1,2] | Wei Yang[2] | Zheng Li*[2] | Yijun Yu[3]

[1]Henan Industrial Technology Academy of Spatio-Temporal Big Data, Henan University, Kaifeng, China
[2]Henan Key Laboratory of Big Data Analysis and Processing, Henan University, Kaifeng, China
[3]Center of Research in Computing, The Open University, Milton Keynes, UK

Correspondence
*Zheng Li, Henan Key Laboratory of Big Data Analysis and Processing, Henan University, Kaifeng 475001, China. Email: lizheng@henu.edu.cn

Summary

With lots of public software descriptions emerging in the application market, it is significant to extract common software features from these descriptions and recommend them to new designers. However, existing approaches often recommend features according to their frequencies which reflect designers' preferences. In order to identify those users' favorite features and help design more popular software, this paper proposes to make use of the public data of users' ratings and products' downloads which reflect users' preferences to recommend extracted features. The proposed approach distinguishes users' perspective from designers' perspective and argues that users' perspective is better for recommending features because most products are designed for users and expect to be popular among users. Based on the lasso regression to estimate the relationship between the extracted features and the users' ratings, it proposes to first distinguish the extracted features to identify those recommendable and undesirable features. By treating each download as a support from users to the product features, it further mines the feature association rules from users' perspective for recommending features. By taking the public data on the market of SoftPedia.com for evaluation, our empirical studies indicate that: (1) selecting recommendable features by lasso regression is better than that by feature frequencies in terms of F_1 *measure*; and (2)recommending features based on the feature association rules mined from users' perspective is not only feasible but also has competitive performance compared with that based on the rules mined from designs' perspective in terms of F_1 *measure*.

KEYWORDS:
feature extraction, feature recommendation, association rules, lasso regression

1 | INTRODUCTION

Requirements analysis is an important basis for software development. However, it is not easy for designers to gain actual user requirements in practice, especially for the software systems used by the mass. Thus, it is beneficial for new designers to analyze the features of the software products in the market. According to Kano model[1], some features are taken for granted by users for all the products in the same application domain. Analyzing the features of the software in the same domain help designers to learn about these common features which users take for granted.

Because software products often rely on specific operating environments and have various competitors in the market such as Google Play and Softpedia.com, it will be exceedingly time-consuming and laborious to analyze the features of existing products manually[2]. On the other hand, lots of software data including natural language product descriptions, user comments, users' ratings and products' downloads have been accumulated and can be easily obtained from the software markets. Consequently, utilizing these public data to identify the common features and recommend them to new designers is appreciated.

Therefore, mining these public software data has gained wide attention in recent years. Various approaches have been proposed for extracting software features from these natural language product descriptions or user comments. For example, in light that more than one feature can be described in one sentence of the product descriptions, Hariri et al.[3] have designed the method of incremental diffusive clustering which clusters the sentences and selects the best clusters iteratively. Through manually identifying the feature patterns from a sampled dataset, Liu et al.[4] have proposed to use them to extract features from app descriptions. To extract features from user reviews, Guzman and Maalej[5] have taken the bigram collocations as feature patterns and identified the frequent collocations from reviews.

However, when recommending these extracted features to new designers, existing approaches often conduct the work from designers' perspective[3,6]. For example, the feature association rules which are mined according to the number of the software that support the features, i.e., whether the associated features appear together frequently in many products, are often used for recommending these extracted features. These association rules only reveal designers' preferences for the feature associations. Since most of software products are designed for users and expected to be popular among users, it is not enough to recommend features with the feature association rules.

In addition, existing approaches often take all the extracted features into consideration without further distinguishing them while recommending, so analysts are required to estimate the number of the implied features accurately when extracting them. Otherwise, some noisy features may be extracted and recommended. But it is difficult for analysts to obtain the accurate number in practice. Some approaches have filtered features according to feature frequencies. However, it is also not suitable because feature frequencies also reflect designers' preferences and some users' favorite features may be infrequent.

In this paper, we propose to recommend features from users' perspective. That is, users' favorite features are identified and recommended for designers according to the public data which reflect users' preferences such as users' ratings and products' downloads. Our main contributions are as follows.

- The users' perspective and the designers' perspective are distinguished for recommending features. We argue that the users' perspective is better since the software systems are designed for users and expect to be popular among users. Recommending features from users' perspective help find users' favorite features.

- According to the data of users' ratings, the lasso regression[7] is applied to distinguish the extracted features before recommending them in order to identify the recommendable and undesirable features. Lasso regression approach can estimate the relationship model between the extracted features and the users' ratings. According to the estimated model, the features positively correlated with the users' ratings are treated as the recommendable features. At the same time the features negatively correlated with users' ratings are taken as the undesirable features. Comparing with multiple linear regression method, lasso regression can compress the estimated model[7] and filter those features which are less likely to correlate with users' ratings. Our studies have found there exists collinearity between the extracted features when the number of them is large. This also means that lasso regression which can address collinearity is preferable to the multiple linear regression.

- According to the data of products' downloads, the feature association rules are mined. Because each download should be treated as a support of users to the product's features, the feature association rules mined can reveal the associations of the recommendable features in the eyes of users. When there are many features recommendable, these association rules help find the users' favorite feature combinations and thus can be used for recommending features from users' perspective.

- By taking the public data of the software of *antivirus* and *compress tool* from SoftPedia.com and extracting bigram collocations as features from the software descriptions, several empirical studies have been carried out. The results indicate that: (1) selecting the recommendable features from these extracted ones by lasso regression is better than that by feature frequencies in terms of F_1 *measure*[8]; and (2) mining the feature association rules from users' perspective by utilizing the data of products' downloads is feasible and according to the F_1 *measure*, recommending features based on those rules has competitive performance compared with that based on the rules mined from designs' perspective.

The remainder of the paper is organized as follows. Section 2 provides a general overview of our approach, while Section 3 describes three steps of the approach, namely, distinguish features with lasso regression, mine feature association rules from

users' perspective and recommend features. Section 4 describes the empirical evaluation design, the results of the evaluation, and the possible limitations and threats to validity. Finally section 5 describes the related work, and Section 6 provides the conclusion of our work and the ideas for future work.

2 | OVERVIEW

Generally, the task of extracting and recommending software features can be divided into two stages as shown in Fig.1. The first stage is to extract common software features from public natural language product data. The second stage is to recommend features to designers. In this paper, we focus mainly on the second stage. In terms of the first stage, any approach of extracting features can be used and we focus on extracting common software features from the natural language software descriptions. The proposed approach aims at identifying and recommending users' favorite features to designers, and helps them design more popular software.

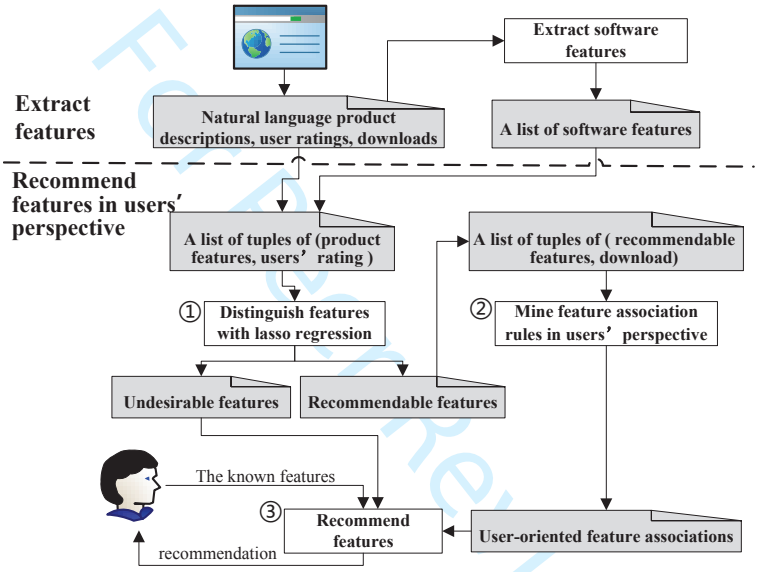


FIGURE 1 The overview of the proposed approach

As can be seen from Fig.1, feature extraction process starts from scraping the public data including natural language product descriptions, users' ratings, and products' downloads from the website such as Softpedia.com. Once all the implied features are extracted, there are three steps in the proposed approach to recommend the features.

- First, the extracted features are distinguished. Through identifying the features possessed by each product and their users' ratings, the list of tuples in the form of (product features, users' rating) is obtained and lasso regression is subsequently used to estimate the relationship model between the extracted features and the users' ratings. According to the estimated model, the features positively and negatively correlated with the users' ratings are then selected. They are the recommendable and undesirable features.

- Second, the association rules revealing the associations of these recommendable features are mined from users' perspective. Through identifying the recommendable features possessed by each product and their downloads, the list of tuples in the form of (recommendable features, download) is obtained and the FP-growth approach[9] is subsequently used to mine the frequent feature combinations. A novel graph is designed to facilitate the use of the association rules implied in these feature combinations.

- Third, given the set of undesirable features and the set of feature association rules, the designers of a new software are advised to check the list of the candidate features and exclude the undesirable features. Subsequently, these recommendable features are recommended to them by applying the feature association rules.

3 | THE APPROACH

In this section, the details of each step of the proposed approach are described.

3.1 | Distinguish features with lasso regression

This step is to distinguish the extracted features before recommending them to designers according to the data of users' ratings. Users' ratings on application markets are often the real numbers. For example, they range from 0 to 5 on SoftPedia.com. Then, linear regression methods can be chose to estimate the relationship model between software features and users' ratings. The coefficients of each feature in the estimated model indicate their importance. Particularly, according to their coefficients, the extracted features can be classified into three categories from the perspective of users.

- ***Recommendable features***: the features have positive coefficients and are recommendable since they are positively correlated with users' ratings.
- ***Optional features***: the features have zero coefficients which indicate they are not correlated with users' ratings.
- ***Undesirable features***: the features have negative coefficients and are not recommendable since they are negatively correlated with users' ratings.

Keertipati, et al. have shown the feasibility of applying the multiple linear regression method [10]. This classification is significant for feature recommendation. Obviously, these undesirable features should be excluded from the list of candidate features and these recommendable features can be reused for new software.

However, there will be too many features positively or negatively correlated with users' ratings and some of them may not be significant ($p > 0.05$) when using the multiple linear regression method. For example, when one hundred features are extracted from the *compress tool* dataset in our evaluation, almost half of them exhibit positive correlation with the users' ratings after applying multiple linear regression method. In this paper, we propose to apply the lasso regression instead of the multiple linear regression with the aim to compress the estimated model and filter those features which are not correlated with users' ratings.

Both multiple linear regression and lasso regression are the methods of linear regression which assume that the relationships between variables can be expressed by a linear function. More formally, let y be a scalar dependent variable, X be the vector of independent variables where $X = [x_1, x_2, \cdots, x_p]$, then multiple linear regression is to estimate the following linear function [11].

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \tag{1}$$

Where $\beta_1, \beta_2, \cdots$ are the coefficients associated with the independent variables. These coefficients including β_0 are often estimated by fitting the above linear function with least squares approach which optimizes the following objective function.

$$min_{\beta_0, \beta^T} \{ \frac{1}{n} \sum_{i}^{n} (y_i - \beta_0 - \beta^T X_i)^2 \} \tag{2}$$

Where $\beta^T = [\beta_1, \beta_2, \cdots, \beta_p]^T$, y_i and X_i are the values of the dependent variable and the independent variable vector for the i-th case.

In practice, when there are many independent variables but the samples for fitting the linear function are not enough, or there exists collinearity among these variables, it is not easy to exactly fit the linear function shown in formula (1). Overfitting often occurs. In this case, regularization is often adopted to penalize these less important variables and compress the model. Lasso regression [7] is one of the methods with the following objective function.

$$min_{\beta_0, \beta^T} \{ \frac{1}{n} \sum_{i}^{n} (y_i - \beta_0 - \beta^T X_i)^2 + \lambda \sum_{\beta_j \in \beta^T} |\beta_j| \} \tag{3}$$

Where $\lambda \sum_{\beta_j \in \beta^T} |\beta_j|$ is the regularization item and λ is the parameter to control the impact of regularization. This regularization item plays as the penalty to force the coefficients of the less important variables to be 0, and thus some independent variables will be excluded from the model. It is challenging to choose the appropriate amount of regularization (i.e. the value of λ) and the cross validation method is often used for this purpose.

By treating the extracted features as the independent variables and taking the users' rating as the dependent variable, the method of lasso regression is adopted to estimate the relationships between software product features and users' ratings. These features with positive and negative coefficients in the fitted linear function are finally selected for recommendations.

In detail, the vector $p_i = (t_{i1}, t_{i2}, \cdots,)$ for each product is generated initially. Given the set of extracted features $\{f_i, f_2, \cdots\}$, if product i has the feature f_j , then $t_{ij} = 1$; otherwise, $t_{ij} = 0$. Then the users' rating is treated as the dependent variable. In this way, a list of tuples (p_i, r_i) is obtained where r_i is the users' rating of product i which is located in the range [0, 5] on Softpedia website. The ratings are not available for some products. To fill the missing users' ratings, the ratings of the most similar products is adopted by calculating the Pearson correlation between p_i. Such list of data (p_i, r_i) is finally input into the lasso regression method to get the coefficients of all extracted features. The implementation of the $lassocv$ which uses the cross validation method to estimate the value of regularization from python package of $sklearn$[12] is used for lasso regression.

3.2 | Mine feature association rules from users' perspective

Once the features with positive and negative coefficients are selected in above step, the product feature vectors p_i will be reduced in size by keeping only those recommendable features. Then, the list of tuples (p'_i, d_i) is built where p'_i is the reduced feature vector of i-th product and d_i is the download of i-th product. Given the list of tuples (p'_i, d_i), this step is to mine the association rules between these recommendable features from users' perspective.

An association rule is the rule in the form $t_1 \rightarrow t_2$ which expresses that when t_1 is observed, t_2 will also appear in some degree[13]. So an association rule indicates the association between two sets of items. This association is often revealed through observing the co-occurrence of the items. Therefore, association rule mining starts from finding these sets of items which occur frequently, named frequent itemsets mining. For an itemset (t_1, t_2, t_3) , if the number of its occurrence is bigger than a value specified by analysts, it is regarded as the frequent itemset. Its occurrence number is called the support, and the value specified by analysts is called the minimal support. Such a frequent itemset means that the items t_1, t_2, t_3 often appear together and thus there exist strong associations between them. Then the association rules such as $t_1 \rightarrow t_2, t_3$ and $t_1, t_2 \rightarrow t_3$ can be derived. To measure the strength of the associations expressed by association rules, the confidence of a rule is used. Taking the rule $t_1 \rightarrow t_2, t_3$ as example, the confidence of this association rule is computed as follows.

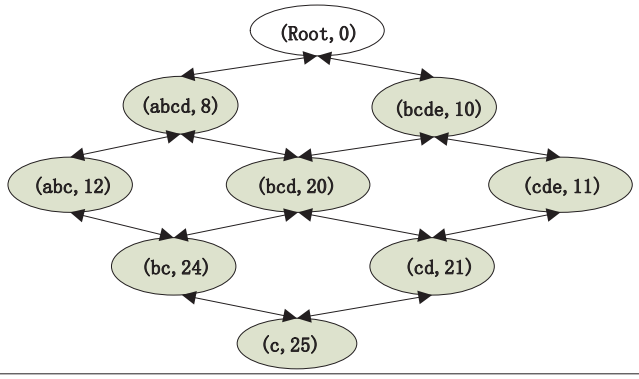
$$conf(t_1 \rightarrow t_2, t_3) = \frac{supp(t_1, t_2, t_3)}{supp(t_1)} \tag{4}$$

The associations implied in association rules can be used for recommendation. That is, if it is known that some items are users' favorites, then the items associated with them can be recommended to the users. In the previous work[3], association rules have been mined and used for feature recommendation. The associations between features are also identified by observing whether a set of features appears together frequently in products. However, different from previous work, two perspectives are distinguished from which the feature association rules can be mined.

- ***Designers' perspective***: the support of a feature set indicates how many products contain all the features in the set.
- ***Users' perspective***: the support of a feature set indicates how many users like the features in the set at the same time.

Obviously, the user-oriented association rules are preferable compared to the designer-oriented association rules for feature recommendation, since most products are designed for users and expect to be popular among the users. Thus, we propose to take the products' downloads into consideration and mine user-oriented association rules by making the assumption that each products' download is a support from users for the recommendable features of the product.

Given the list of data in the form (p'_i, d_i), we apply the FP-growth algorithm[9] to mine the frequent feature sets. To facilitate the reuse of association rules, we design a compressed frequent feature set graph which looks like a tree shown in Fig. 2 to store the mined frequent feature sets. In such a graph, each node stores a frequent feature set and its support. The association rules are implied in the connections between nodes. For example, the connection between node $(c, 25)$ and $(cd, 21)$ implies the rule $c \rightarrow d$ with the confidence of 0.84, i.e., $21/25$.



<Feature set, download>:	<Frequent feature sets, support>:(min-support=2)			
1. <abcd, 8>	1.<e,11>	7.<ed,11>	13.<adbc,8>	19.<dbc,20>
2. <bcde,10>	2.<eb,10>	8.<edc,11>	14.<ac,12>	20.<dc,21>
3. <bcd, 2>	3.<ebc,10>	9.<a,12>	15.<ab,12>	21.<b,24>
4. <abc, 4>	4.<ebd,10>	10.<ad,8>	16.<abc,12>	22.<bc,24>
5. <cde, 1>	5.<ebdc,10>	11.<adc,8>	17.<d,21>	23.<c,25>
	6.<ec,11>	12.<adb,8>	18.<db,20>	

FIGURE 2 The frequent feature set graph.

The similar data structure has been used in the work of Hariri et al.[3] and Yu et al.[6] for the same purpose. However, the frequent feature set graph here is a compressed and inverted one. By compression we mean that for a frequent feature set, if there exists a frequent feature set which not only contains it but also has the same support, then it will be not stored in the graph. It is to avoid building a very large graph caused by the set explosion when mining frequent feature sets. Because all the subsets of a frequent feature set are also frequent feature sets, a large number of frequent feature sets will be generated when mining feature sets based on FP-growth algorithm. For example, as can be seen in the bottom of Fig. 2, 23 frequent feature sets have been generated given the data from 5 products. And obviously, some feature sets have the same supports with their parents, i.e., the feature sets which contain them. For these feature sets, none information will be gained when storing them in the graph.

Therefore, the frequent feature set graph shown in Fig. 2 is designed as an inverted tree. That is, there are different levels for the nodes in the graph and the ancestors are in the higher levels. The frequent feature sets which have the same supports with their parents can be filtered when building the graph. Initially, the frequent feature sets returned from FP-growth algorithm are sorted from large to small according to their size. This makes all the parent sets of a feature set appearing before him. Then these frequent feature sets are inserted into the graph one by one in that order. A recursive algorithm has been designed for the insertion, which can be seen in algorithm 1. Before insertion, a new node will be built initially for a new feature set. Starting from the root node of the graph, the algorithm evaluates the node under consideration and finds all its child nodes which contain the new feature set (i.e., the 2nd step in the algorithm). Once none child nodes are found, the new node is inserted as a child node of the node under consideration (i.e, the 4th, 5th, and 6th steps). Otherwise, all the child nodes which are found are evaluated again to filter these nodes which have the same support with the new node (i.e., 3rd step). If there are some nodes left, each of them is taken into consideration while repeating above process (i.e., 7th, 8th, 9th and 10th steps).

During feature recommendation, the association rules are produced and applied online. Given a feature set, if it is located in the graph, all the sibling features in its ancestors are recommended when the confidence is larger than a threshold, i.e., the $min\_confidence$. Otherwise, the parent node which contains it will be found, and similarly the sibling features in both parent node and parent node's ancestors are recommended. For example, when feature c in Fig. 2 is given, it is located in the graph. Then, the ancestor nodes bc, cd and bcd are found when we set $min\_confidence = 0.8$. And thus the sibling features b and d are recommended. When feature set bd is given, it is not located in the graph and its parent node bcd will be first found. Then only the sibling feature c is recommended with $confidence = 1$. Even if there are ancestor nodes $abcd$ and $bcde$, they are not taken into consideration since the $confidence < 0.8$.

Algorithm 1 inserting feature set

Function *insertFeatSet (root_node,new_set,new_node)*

Data:
root_node: the root node of the inverted tree
new_set: the feature set to be inserted
new_node: the node built for the *new_set*
Result:
The boolean result indicating whether the *new_node* is inserted

```
1  begin
2      D ← findChildren(root_node, new_set) //find all the child nodes of root_node that contain new_set
3      E ← filterNodes(D, new_node) //filter all the children with the same support with new_set
4      if |D| = 0 then
5          insertChild(root_node, new_node)//insert the new_node as a child of root_node  return True
6      else if |E| ≠ 0 then
7          result ← False  for each node ∈ E do
8              result ← result∨ insertFeatSet(node, new_set, new_node)
9          end
10         return result
11     return False
12 end
```

3.3 | Recommend features

Given the set of undesirable features and the set of user-oriented association rules which reflect the associations between these recommendable features, this step is to advise designers to include or exclude some features. Firstly, those undesirable features are proposed to be excluded from the list of candidate features. Secondly, these recommendable features are recommended to designers by applying the set of user-oriented feature association rules.

4 | EMPIRICAL EVALUATION

To evaluate our approach, several empirical studies have been done. In this section, we introduce the research questions, the study design and results, and the potential threats to validity.

4.1 | Research questions(RQs)

Our evaluations aim to answer the following four RQs:

RQ1: *How about applying lasso regression to distinguish extracted features?* Since some noisy features may be extracted, it is beneficial to distinguish these extracted features and select these recommendable ones for recommendations. Given the data of user ratings, our approach applies lasso regression to conduct this task from users' perspective. RQ1 aims to observe the effectiveness of lasso regression to distinguish the extracted features.

RQ2: *How about recommending features based on user-oriented association rules?* This paper proposes to mine the feature association rules from the users' perspective for recommending features. RQ2 aims to evaluate the performance of recommending features based on user-oriented association rules by comparing it with recommending features based on designer-oriented association rules.

4.2 | Study design

4.2.1 | Data

To answer the research questions, we have taken the public software data of *antivirus* and *compress tool* from Softpedia.com scraped in October 2018 for evaluations. The data include the natural language descriptions about the software products, the

users' ratings and the products' downloads. The natural language descriptions for each product are under the label of summary. As shown in Table 1, there are 576 software products in *antivirus* category and 400 software products in *compress tool* category after eliminating the duplicate ones and the flawed ones which lack the descriptions. All the data are available at github (https://github.com/UPFR/ERF).

Because the proposed approach depends on extracting software features from these natural language product descriptions and any approach can be used for this purpose, we have taken bigram collocation based approach used in the work of Guzman and Maalej[5] to extract software features. First, these natural language product descriptions are parsed into sentences. Then, based on the tool of NLTK[14], each sentence is tokenized into a set of words; the words are marked with their part of speech; only the verbs, nouns, and adjectives are kept; stop words are removed; and the left words are stemmed to their root forms. The sentences consisting of the same words for the same product are also eliminated. After that, the bigram collocations within three words distance are elicited and their frequencies are counted. Further, the bigram collocations are sorted according to their frequency and these most frequent bigram collocations are selected. Considering the possible number of features possessed by the software products in one domain, top 25, 50 and 100 frequent bigram collocations are selected in our evaluations. Finally, the synonym collocations among these selected ones are grouped together based on WordNet[15]. Each group of bigram collocations is treated as one feature. All the implementations can also be found in the github.

TABLE 1 The datasets for evaluation

Dataset	Num. software products
Antivirus	576
Compress tool	400

4.2.2 | Descriptions of studies

To answer the research questions, three studies have been designed.

- ***Study I****: lasso regression vs. multiple linear regression*.

This study is to test the effectiveness of lasso regression to distinguish extracted features by comparing it with multiple regression. We first studied the collinearity problem and investigated whether the extracted features were correlated with each other. This will show the reasonability of applying the lasso regression. In regression when several independent variables are highly correlated, this problem is called multi-collinearity or collinearity[11]. It makes the estimated model distorted, and can be addressed by lasso regression through compressing the estimated model. Collinearity is often measured by variation inflation factor (vif). Variation inflation is the consequence of collinearity. A high vif indicates that the associated independent variable is highly collinear with the other variables in the model. It is often considered that the collinearity exists when $vif > 2.5$[11].

With the *antivirus* and *compress tool* datasets, we further used the multiple linear regression and the lasso regression to estimate the linear relationships between the extracted software features and the users' ratings. By comparing with the multiple regression, we observed the effectiveness of lasso regression to distinguish extracted features.

- ***Study II****: lasso regression vs. feature frequency*.

This study is to further evaluate the effectiveness of lasso regression on selecting the recommendable features by comparing it with that of selecting features according to feature frequencies. For this purpose, the datasets of *antivirus* and *compress tool* were first sampled and 100 product descriptions were selected for manual analysis to identify the implied software features. These identified features are the answers for evaluations. Two researchers in our group who were familiar with the *antivirus* and *compress tool* applications were required to first identify the software features from each dataset independently and discuss their results subsequently to produce the answers. For each feature, a set of representative phrases were selected as the feature descriptors. The answer sets of software features of these two datasets are also available at github.

For comparison, the same number of features by two methods were selected, and F_1 *measure* defined as follows was used[8].

$$F_1 = \frac{2PR}{(P+R)} \tag{5}$$

P is the precision and R is the recall. Let I be the set of features selecting according to feature frequencies, L be the set of recommendable features selected by lasso regression, and A be the set of features identified manually, then for selecting recommendable features by lasso regression, $P = |(L \cap A)|/|L|$ and $R = |(L \cap A)|/|A|$, and for selecting features by feature frequencies, $P = |(I \cap A)|/|I|$ and $R = |(I \cap A)|/|A|$.

- ***Study III****: performance of recommending features based on user-oriented association rules.*

This paper proposes to mine the feature association rules for recommending features from the users' perspective, i.e., according to the number of users who support the features, instead of the designers' perspective, i.e., according to the number of software which contain the features. This study is to evaluate the different performance of recommending features based on user-oriented association rules and designer-oriented association rules. The m-fold cross validation approach was adopted which is often used for recommendation system evaluations [16]. In this approach, the dataset is divided into m parts and there are m runs in the experiments. In each run, one part of the dataset is selected as the test set and all the remaining parts are taken as the training sets. The average performance of the m runs is finally taken as the performance of the recommendation system.

We set $m = 5$ in our cross validation experiments. That is, the list of data (p'_i, d_i) is divided into 5 parts where p'_i is the set of recommendable features of i-th product and d_i is the download of i-th product. When mining designer-oriented feature association rules, $d_i = 1$. In each run of the experiments, four parts of data were used to mine frequent feature sets and generate the association rules and the left part was used for testing the recommendation performance of applying the generated association rules. In one recommendation test, two features were selected from the feature set of one product in the test set as the initial product features. The remaining features of the product were the targets of recommendation.

To calculate the performance in one recommendation test, F_1 *measure* which is defined in the formula (5) was also used [8]. Differently, P here is the recommendation precision and R is the recommendation recall. Let T be the target feature set including the remaining features of the product, and H be the feature set including the features recommended by applying the association rules, then P and R are calculated as follows.

$$P = \frac{|T \bigcap H|}{|H|} \tag{6}$$

$$R = \frac{|T \bigcap H|}{|T|} \tag{7}$$

For each product in test set, all two-member feature subsets were generated for test and the average performance was taken as the recommendation performance of the product. Finally, the average recommendation performance of all the products in the test set was taken as the recommendation performance in one run. Considering the usual values of the $min\_confidence$, we set $min\_confidence = 0.5, 0.6, 0.7$ and 0.8 in the experiment. And to facilitate setting the minimum support in frequent feature sets mining, it was assumed that each product represented a designer and each download represented a user. Then, different ratios was adopted to generate different minimum supports for mining the user-oriented association rules and the designer-oriented association rules. That is, the same ratio was multiplied with the number of products in the training sets and the total number of downloads of the products in the training sets to generate the different minimum supports. And the recommendation performance based on the user-oriented association rules and the designer-oriented association rules were compared under the same ratio.

4.3 | Results

4.3.1 | RQ1: How about applying lasso regression to distinguish extracted features?

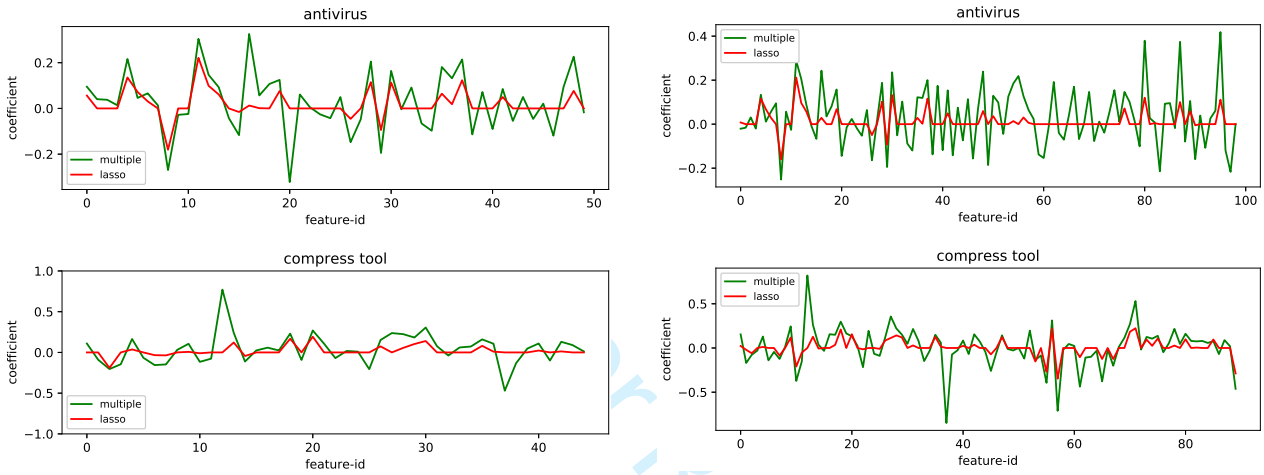
Table 2 shows the number of extracted features that are considered to be highly correlated with other features. The level means the number of frequent bigram collocations selected for evaluation. It can be observed that there exists collinearity between these extracted features for both datasets. And when more features are extracted, there are more features correlated with each other. This indicates that it is beneficial for applying the lasso regression method which can address the collinearity problem.

With the *antivirus* and *compress tool* datasets, Fig. 3 shows the coefficients of different features when applying both methods. It can be seen that while keeping the important features, lasso regression forces the coefficients of many features which are less likely to correlate with users' ratings to be 0. More exactly, Table 3 shows the number of features with positive coefficients by different methods. It is obvious that lasso regression can reduce the features positively correlated with the users' ratings. For example, there are 26 features with positive coefficients from *antivirus* dataset when selecting top 100 frequent bigram collocations for evaluation. At the same time, there are 60 features by multiple linear regression. Also, when selecting top 100

TABLE 2 The collinearity between extracted features

Level	Dataset	Num. features with $vif > 2.5$
50	Antivirus	4
	Compress tool	2
100	Antivirus	11
	Compress tool	13

frequent bigram collocations for evaluation from *compress tool* dataset, lasso regression identifies 27 features with positive coefficients. At the same time, there are 50 features by multiple linear regression.



(a) selecting top 50 frequent collocations for evaluation

(b) selecting top 100 frequent collocations for evaluation

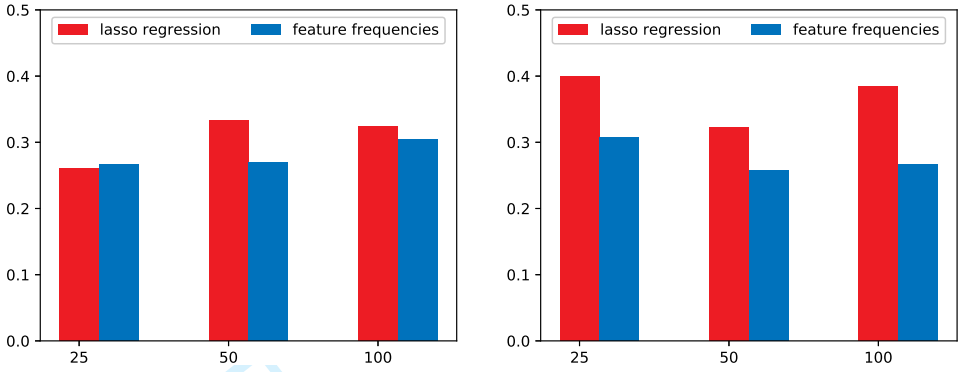
FIGURE 3 The regression results

TABLE 3 The number of features with positive coefficients

Level	Dataset	multiple linear regression	lasso regression
50	Antivirus	30	17
	Compress tool	24	13
100	Antivirus	60	26
	Compress tool	50	27

Fig.4 shows the comparison results about selecting features by lasso regression and feature frequencies. It can be seen that selecting recommendable features by lasso regression performs better in terms of F_1 *measure*. We have further checked the features selected by two different methods to investigate their difference. When selecting top 100 bigram collocations as extracted features for evaluation, Table 4 shows the top 10 recommendable features with larger coefficients after applying lasso regression(the places of each feature in the rank of frequency are also given) and the top 10 recommendable features with higher frequency. It can be seen that the approach based on lasso regression can select features with richer diversity. The features

with low frequency such as *archive type* and *guard time* can also be selected. Instead, many features selected by feature frequency are concerned about the same subjects, for example, *virus antivirus*, *virus anti*, *remove virus* and *scanner virus*. This phenomenon may be due to the fact that different designers tend to use different phrases to express the same popular features.



(a) *antivirus* (b) *compress tool*

FIGURE 4 The comparison about selecting features by lasso regression and feature frequencies in terms of F_1 *measure*

TABLE 4 The top 10 recommendable features selected by lasso regression and by feature frequency

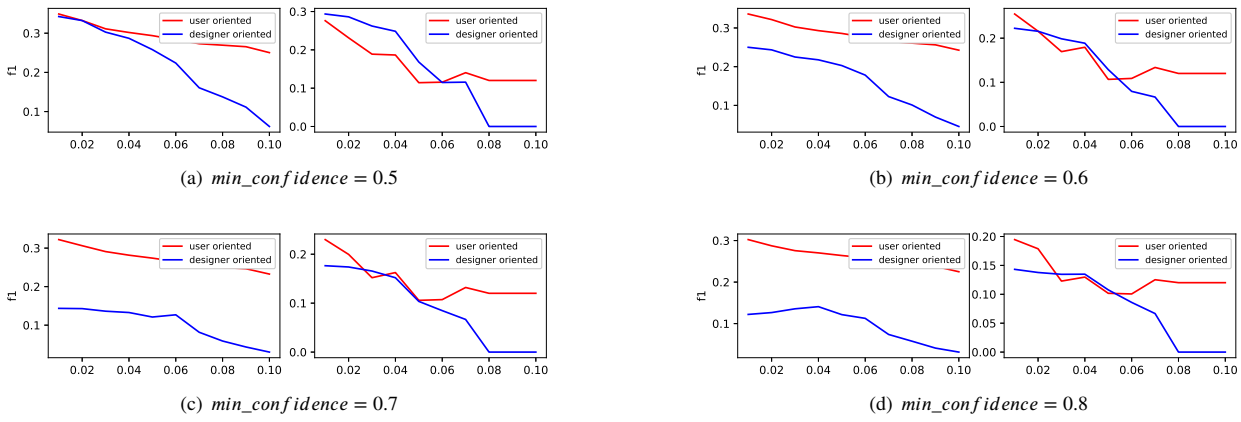
Antivirus		Compress tool	
Features by lasso	**Features by frequency**	**Features by lasso**	**Features by frequency**
anti malware(11)	real time	archiver compress(71)	file archive
solution security(30)	antivirus protection	file new(56)	file compress
operate system(80)	virus antivirus	tool compression(18)	zip file
internet security(4)	scanner virus	archive type(70)	file extract
real protection(20)	internet security	menu context(20)	archive create
guard time(95)	malware antivirus	line command(28)	archive extract
perform scan(28)	virus malware	zip rar(13)	compress archive
antivirus antispyware(87)	virus trojan	main window(35)	zip archive
scan file(12)	virus anti	level compression(47)	compression file
antivirus application(76)	remove virus	zip format(29)	file folder

4.3.2 | RQ2: How about recommending features based on user-oriented association rules?

When selecting different numbers of frequent bigram collocations as features, Fig. 5 and 6 show the performance of feature recommendation based on user-oriented association rules and designer-oriented association rules in terms of precision, recall and F_1. It can be seen that according to the performance in F_1, the recommendations based on user-oriented association rules have better performance in most cases, especially when 100 features are selected for evaluation.

4.4 | Limitations and threats to validity

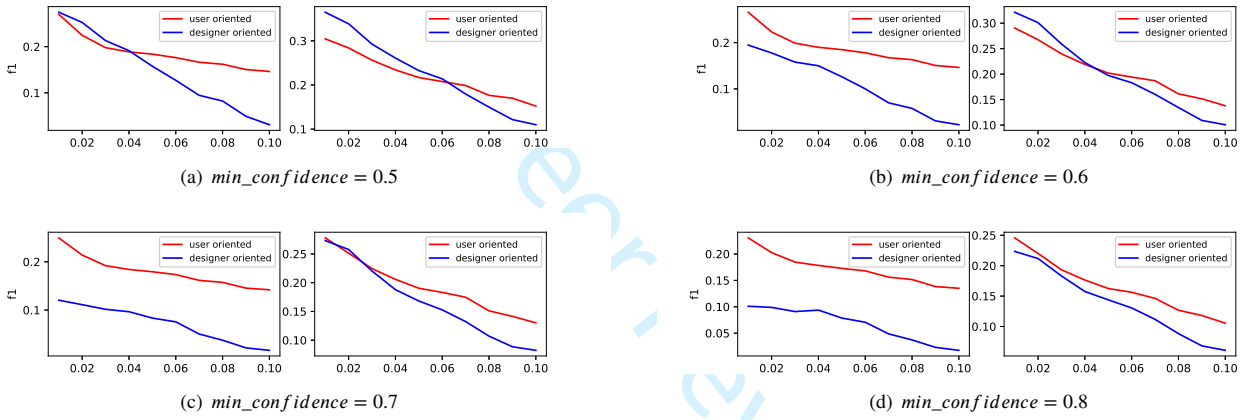
Limitation. Although we have shown the feasibility and benefits of recommending features from users' perspective, it should be pointed out that the result of the proposed approach is limited by the result of feature extraction. From Table 3, it can be seen that the number of recommendable and undesirable features selected from the extracted features depends on the number of extracted



(a) $min\_confidence = 0.5$ (b) $min\_confidence = 0.6$

(c) $min\_confidence = 0.7$ (d) $min\_confidence = 0.8$

FIGURE 5 The comparison when selecting top 50 frequent bigram collocations for evaluation



(a) $min\_confidence = 0.5$ (b) $min\_confidence = 0.6$

(c) $min\_confidence = 0.7$ (d) $min\_confidence = 0.8$

FIGURE 6 The comparison when selecting top 100 frequent bigram collocations for evaluation

features. Particularly, our approach is more suitable for the case that the analysts have no ideas about the exact number of the features and a larger number of features are extracted at last.

Internal validity. First, to distinguish the extracted features and identify those recommendable and undesirable ones among them, we propose to make use of the data of users' ratings and apply the lasso regression to estimate the relationships between those extracted features and the users' ratings. It should be pointed out that the ratings of the products may not reflect the users' preferences on product features. Users may not have taken the ratings they give seriously. Their ratings may also be affected by the factors such as the reputation of the product designers and the publishing time of the products. Consequently, these may affect the results of feature selection based on lasso regression. Fortunately, the users' ratings we obtain from the application markets are the combinations of the ratings from many users based on the invisible hands of the markets. This can reduce the potential deviation to some extent.

Second, to validate the effectiveness of selecting recommendable features based on lasso regression method, it needs to survey a number of users and find out what the recommendable features are for comparison. Obviously it is not only laborious but also expensive. We have had two researchers in our group to identify the features from a random sample of 100 product descriptions. The two researchers may not represent the users of these software systems and the sampled product descriptions may not represent all the product descriptions in the datasets.

Third, evaluating a recommendation system is a non-trivial task. The ideal way is to design a product with the recommended features and observe its performance in the market. Obviously it will cost a long time. In order to compare the recommendation performance based on user-oriented association rules and designer-oriented association rules, we have assumed that each product

represents a designer and each download represents a user. Obviously, it may not be the case in practice. That is, one designer may design more than one product, and each user can download more than one product. Further, we have assumed that each download represents a support from users to the set of features of a product. It may also not be true because the users may download a product because of other reasons except the product features. All these may affect the validity of the evaluation on the recommendation performance of the user-oriented association rules.

External validity. We applied our approach to the public data of two categories of software on SoftPedia.com. In terms of the performance of distinguishing extracted features by lasso regression, the results are shown with consistency. In terms of feature recommendation based on user-oriented association rules, the results are also shown with consistency when many features are extracted. These provide confidence about the generalizability of our results. Further experiments with the public data of other categories of software from SoftPedia.com or from other software markets would nevertheless be useful for improving external validity.

5 | RELATED WORK

As lots of software data have been accumulated on the application markets such as GooglePlay and SoftPedia, mining these data has gained wide attention in recent years. In this section, we outline the most related work on extracting software features from these data and recommending them to designers.

Because textual product descriptions mining can build the domain feature model [17,18] and user comments mining can identify new features desired by users, extracting software features from these textual data is of great interest to many researchers [19,20,21]. One kind of approaches tends to cluster the sentences of the textual data into different clusters by different methods. Each sentence cluster implies one software feature. For example, Hariri et al. [3] have proposed an incremental diffusive clustering (IDC) algorithm to obtain the sentence clusters. Bakar et al. [22] have used the SVD method to get the sentence clusters. Yu et al. [6] have taken LDA(Latent dirichlet allocation) method to turn the sentences into vectors and then applied a hierarchical clustering method to cluster the sentences with the aim to mine the semantic structure between features. Recently, Li et al. [23] intended to extract features with the convolutional neural network(CNN).

Considering that features are often expressed in the form of phrases, i.e., a sequence of words, another kind of approaches tends to first define the patterns of the phrases, then extract phrases from each sentence, and finally group them to merge synonym ones. For example, Vu et al. [24,25] have first mined the sequence pattern of part of speech (POS) from a corpus and then used them to extract features from app reviews. Liu et al. [4] have obtained the patterns of phrases manually from a sampled dataset in order to extract features from app descriptions. Aiming at extracting and matching the features from the app descriptions and the reviews, Johann et al. [26] have also manually built several patterns and applied them to app descriptions and user reviews. Unlike aforementioned work, Sarro et al. have first extracted raw feature patterns [27] to identify the app features and analyze feature lifecycle in app stores. In their work, if an HTML list in the description of apps was related with the set of words "include, new, latest, key, free, improved, download, option, feature", the HTML list was saved as the raw feature patterns. Moreover, Guzman and Maalej [5] have directly taken bigram collocations as feature pattern to extract features from user reviews.

While there are many works on feature extraction, little attention has been paid on distinguishing the extracted features and identifying those more valuable features. One similar work is that of Keertipati et al [10]. By taking the data such as frequency and ratings, they have proposed several methods including the multiple linear regression to prioritize the features extracted from app reviews. Different from this work, our purpose is to select those recommendable and undesirable features from users' perspective from these extracted ones instead of prioritizing them. Further, we have used the lasso regression instead of the multiple linear regression to filter those features that are less likely to correlate with users' ratings.

For the task of feature recommendation, these recommendation techniques such as k nearest neighbor (KNN) [3] and association rules [3,6] have been used. When applying association rules for feature recommendation, the existing approaches have mined the feature association rules from designers' perspective. Different from existing works, we distinguish mining association rules from users' perspective from that from designers' perspective by treating each download of a product as a support from users for the set of product features. We propose that the user-oriented association rules are preferable to the designer-oriented association rules because they reveal the associations between features in the eyes of users.

6 | CONCLUSION AND FUTURE WORK

Focusing on extracting and recommending common software features based on the public data on the software markets, this paper has shown how to use the public data of users' ratings and products' downloads which reflect users' preferences for recommending features.

It proposes to distinguish the extracted features to identify the recommendable and undesirable features before recommending them to designers. While recommending these recommendable features to designers, it advises designers to exclude these undesirable features. To distinguish the extracted features from users' perspective, the lasso regression is applied to estimate the relationship between the extracted features and the users' ratings. For recommending features from users' perspective, the user-oriented feature association rules are mined by assuming that each download is a support from users to the product features. By taking the data on the market of SoftPedia.com for evaluation, our empirical studies have confirmed that lasso regression is preferable to the multiple linear regression for estimating the relationship between the product features and the users' ratings. Lasso regression can address the collinearity which is found between the extracted features. It can also filter those features which are less likely to correlate with users' ratings. Through identifying the software features manually for comparison, our studies indicate that selecting the recommendable features from these extracted ones by lasso regression is better than that by feature frequencies in terms of F_1 *measure*. Our studies have further indicate that mining the feature association rules from users' perspective is feasible by utilizing the data of products' downloads. And according to the F_1 *measure*, recommending features based on those rules has competitive performance compared with that based on the rules mined from designs' perspective.

Moreover, the proposed approach has only used the positive feature correlations in the eyes of users for recommending features. In fact, there are also negative correlations and it is better to combine the positive and negative correlations for feature recommendations. Besides the users' ratings and product downloads, there are other users' data such as reviews that can be used for recommending features from users' perspective. Further, the extracted features may be similar with each other and it is better to merge them together before recommending them. In the future, we will pay more efforts to investigate and address above issues.

ACKNOWLEDGMENTS

This work is partially supported by NSFC under Grant No.61402150,61806074.

References

1. Berger C, Blauth R, Boger D, Bolster C. Kano's methods for understanding customer-defined quality. *Center for Quality Management Journal* 1993; 2(4): 3–36. Available: http://walden-family.com/public/cqm-journal/2-4-Whole-Issue.pdf.

2. Ferrari A, Spagnolo G, Dell'Orletta F. Mining commonalities and variabilities from natural language documents. In: 17th International Conference on Software Product Line. ; 2013; Tokyo, Japan.

3. Hariri N, Castro-Herrera C, Mirakhorli M, Cleland-Huang J, Mobasher B. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Transactions on Software Engineering* 2013; 39(12): 1736–1752.

4. Liu Y, Liu L, Liu H, Wang X, Yang H. Mining domain knowledge from app descriptions. *Journal of Systems and Software* 2017; 133: 126–144.

5. Guzman E, Maalej W. How do users like this feature? a fine grained sentiment analysis of app reviews. In: Proc. IEEE International Requirements Engineering Conference(RE'14). ; 2014; Karlskrona, Sweden.

6. Yu Y, Wang H, Yin G, Liu B. Mining and recommending software features across multiple web repositories. In: Proc. 5th Asia-Pacific Symposium on Internetware. ; 2013; Changsha, China. Article No. 9.

7. Tibshirani R. Regression Shrinkage and Selection via the lasso. *Journal of the Royal Statistical Society: Series B* 1996; 58(1): 267–288.

8. Manning C, Raghavan P, Schutze H. *Introduction to information retrieval,1 edition*. Cambridge University Press . 2008. Available: https://nlp.stanford.edu/IR-book/pdf/08eval.pdf.

9. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD '00). ; 2000; Dallas, Texas, USA.

10. Keertipati S, Savarimuthu B, Licorish S. Approaches for prioritizing feature improvements extracted from app reviews. In: Proc. 20th International Conference on Evaluation and Assessment in Software Engineering(EASE'16). ; 2016; Limerick, Ireland.

11. Allison P. *Multiple Regression: A Primer,1st edition*. Pine Forge Press . 1999. Available: http://www.doc88.com/p-9495699527826.html.

12. Scikit-learn:machine learning in python. *https://scikit-learn.org/stable/*.

13. Agrawal R, Imieli'nski T, Swami A. Mining association rules between sets of items in large databases. In: Proc. ACM International Conference on Management of data (SIGMOD '93). ; 1993; Washington, D.C., USA.

14. Bird S, Klein E, Loper E. *Natural language processing with python (1st ed.)*. O'Reilly Media, Inc. . 2009.

15. Miller G. WordNet: a lexical database for english. *Communications of the ACM* 1995; 38(11): 39–41.

16. Herlocker J, Konstan J, Terveen L, Riedl J. Evaluating collaborative filtering recommender systems. *ACM Transaction on information systems* 2004; 22(1): 5–53.

17. Sree-Kumar A, Planas E, Clariso R. Extracting software product line feature models from natural language specifications. In: Proc. 22nd International Systems and Software Product Line Conference(SPLC'18). ; 2018; Gothenburg, Sweden.

18. Liu Y, Liu L, Liu H, Yin X. App store mining for iterative domain analysis: combine app descriptions with user reviews. *Software: Practice and Experience* 2019; 49(6): 1–28.

19. Iacob C, Harrison R. Retrieving and analyzing mobile apps feature requests from online reviews. In: Proc. 10th Working Conference on Mining Software Repositories (MSR'13). ; 2013; San Francisco, CA, USA.

20. Acher M, Cleve A, Perrouin G, et al. On extracting feature models from product descriptions. In: Proc. Sixth International Workshop on Variability Modeling of Software-Intensive Systems(VaMoS'12). ; 2012; Leipzig, Germany.

21. Bakar N, Kasirun Z, Salleh N. Feature extraction approaches from natural language requirements for reuse in software product lines: a systematic literature review. *Journal of Systems and Software* 2015; 106: 132–149.

22. Bakar N, Kasirun Z, Salleh N, Jalab H. Extracting features from online software reviews to aid requirements reuse. *Applied Soft Computing* 2016; 49: 1297–1315.

23. Li Y, Schulze S, Saake G. Extracting features from requirements:achieving accuracy and automation with neural networks. In: Proc. IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). ; 2018; Campobasso, Italy.

24. Vu P, Pham H, Nguyen T, Nguyen TT. Phrase-based extraction of user opinions in mobile app reviews. In: Proc. 31st IEEE/ACM International Conference on Automated Software Engineering (ASE'16). ; 2016; Singapore.

25. Vu P, Nguyen T, Pham H. Mining user opinions in mobile app reviews: a keyword-based approach. In: Proc. 30th IEEE/ACM International Conference on Automated Software Engineering (ASE'15). ; 2015; Lincoln, NE, USA.

26. Johann T, Stanik C, Alizadeh B. A, Maalej W. SAFE: a simple approach for feature extraction from app descriptions and app reviews. In: Proc. 25th IEEE Internationa Requirements Engineering Conference (RE'17). ; 2017; Lisbon, Portugal.

27. Sarro F, Al-Subaihin A, Harman M, Jia Y, Martin W, Zhang Y. Feature lifecycles as they spread, migrate, remain, and die in app stores. In: Proc. 23rd IEEE International Requirements Engineering Conference (RE'15). ; 2015; Ottawa, ON, Canada.

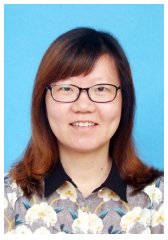
AUTHOR BIOGRAPHY

Chun Liu was born in Xinyang, Henan, China in 1982. He received his Ph.D. degree in software engineering from Institute of Mathematics, Chinese Academy of Sciences, Beijing, China, in 2012.

From 2012 to 2016, he was a lecturer with the School of Computer and Information Engineering, Henan University, Kaifeng, China. Since 2016, he was a assistant professor with Henan Industrial Technology Academy of Spatio-Temporal Big Data and Henan Key Laboratory of Big Data Analysis and Processing, Henan University. From 2014 to 2015, he has been a visiting scholar with the School of Computing and Communications, The Open University, Milton Keynes, UK. His research interests include software engineering and big data.

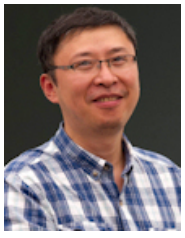
Wei Yang was born in Xinyang, Henan, China in 1983. He received his Ph.D. degree in computer application technology from Harbin Institute of Technology in 2011.

From 2012 to 2017, he was a lecturer with the School of Computer and Information Engineering, Henan University, Kaifeng, China. Since 2017, he was a assistant professor with the Department of Data Science, Henan University. From 2014 to 2015, he has been a visiting scholar with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests are in the areas of machine learning and pattern recognition.



Zheng Li was born in Zhumadian, Henan, China in 1984. She received the Ph.D. degree in computer software and theory from Wuhan University, Wuhan, Hubei, in 2013.

From 2013 to 2017, she was a lecturer of the School of Computer and Information Engineering, Henan University, Kaifeng, Henan, China. Since 2018, she has been an associate professor with the Department of Data Science, Henan University. Her research interests include services computing, software engineering and big data.



Yijun Yu graduated from the Department of Computer Science at Fudan University (B.S. 1992, M.S. 1995, Ph.D. 1998). He was a postdoctoral research fellow at the Department of Electrical Engineering in Ghent University (1999–2002), then he worked as a research associate and lecturer at the Department of Computer Science in University of Toronto (2003-2006). Since October 2006, he has become a senior lecturer at the School of Computing and Communications at The Open University.

His research interests include automated software engineering, requirements engineering, software maintenance and evolution. He serves as an Associate Editor of the Software Quality Journal, Chair of BCS Specialist Group on Requirements Engineering, Program Chair of international workshop on Natural Language for Software Engineering at ESEC FSE'18. He is a PC member of international conferences on Software Engineering (FSE, ICSE), Requirements Engineering (RE, CAiSE,SEAMS), Software Maintenance and Evolution (ICSME, CSMR, SANER, ICPC), Security (ESSoS), and World Forum on Internet of Things (WF-IoT).

He has received a 10 Year Most Influential Paper award (CASCON'16), 4 Best Paper awards (SEAMS'18, iRENICS'16, TrustCom'14, EICS'13), 3 Distinguished Paper awards (RE'11, BCS'08, ASE'07), and a Best Tool Demo Paper (RE'13) award.